

UNITED STATES PATENT APPLICATION

FOR

MAC BUS INTERFACE

Inventors: Gavin Stark
John Wishneusky

Prepared By:
BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN
12400 Wilshire Boulevard
Seventh Floor
Los Angeles, California 90025-1026
(425) 827-8600

Attorney's Docket No.: 042390.P9927

"Express Mail" mailing label number: EL431687449US

Date of Deposit December 28, 2000

I hereby certify that I am causing this paper or fee to be deposited with the United States Postal Service "Express Mail Post Office to Addressee" service on the date indicated above and that this paper or fee has been addressed to the Assistant Commissioner for Patents, Washington, D. C. 20231

Dominique Valentino

(Typed or printed name of person mailing paper or fee)

Dominique Valentino

(Signature of person mailing paper or fee)

12-28-00

(Date signed)

MAC BUS INTERFACE

BACKGROUND OF THE INVENTION

Field of the Invention

5 The present invention generally concerns communications interfaces, and in more particular concerns a scheme for interfacing a plurality of MAC interfaces to a single communications controller.

Background Information

10 The problem of interconnecting a number of physical (layer 1) or Media Access Control (MAC) (layer2) communications interfaces to a single communications controller has been around for years. A number of standard interfaces have been developed to address this problem. These include Utopia1, Utopia2, and Utopia 3 (for ATM systems), and MII and RMII for IEEE802.3 systems. These systems tend to be physical layer or
15 MAC layer dependent, and not well suited to route interfaces onboard an ASIC. For example, the Utopia interfaces only work for ATM cells, and use tristate buses. In addition, the RMII and MII interfaces do not multiplex data for many channels over a single set of wires, but require separate wires for each channel.

20 Because of these limitations, many ASIC designs that incorporate communications interfaces have tended to include internal ASIC buses with an address and a data bus and DMA (direct memory access) to supply the required functionality. The designs have then become very

focused on the particular communications physical or MAC layer that is being utilized, and they have not been generally applicable to the breadth of communications interfaces that the industry uses. This leads to poor reusability of designs, and due to the tight coupling between the DMA
5 engine and the communications interface, the ASIC design process leads to interdependencies between the two, such as routing sensitivity.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing aspects and many of the attendant advantages of this invention will become more readily appreciated as the same becomes better understood by reference to the following detailed description, when
5 taken in conjunction with the accompanying drawings, wherein:

FIGURE 1 is a high-level schematic block diagram illustrating communication between a system side and a network side via a MAC layer and a physical layer;

FIGURE 2 is a schematic block diagram an exemplary
10 configuration that includes a plurality of MAC interfaces in accord with the present invention;

FIGURE 3 is a schematic block diagram illustrating a cascaded pair of MAC bus multiplexors that enable the configuration in FIGURE 2 to operate;

15 FIGURE 4 is a schematic diagram illustrating the MAC Bus In communication signals corresponding to a transfer of data from a network side block to a system side block;

FIGURE 5 is a schematic diagram illustrating the MAC Bus Out communication signals that enable a transfer of data from a system side
20 block to a network side block and enable the network side block to send flow control information messages to the system side block;

FIGURE 6 is a data packet header word used when transferring

data across the MAC bus interface;

FIGURE 7 is a request message word used when sending a message across the MAC bus interface;

FIGURE 8 is a schematic diagram of a MAC Bus In Data

5 Multiplexor;

FIGURE 9 is a schematic diagram of a MAC Bus Out Data

Multiplexor;

FIGURE 10 is a schematic diagram of a MAC Bus Out Message

Multiplexor;

10 FIGURE 11 shows a set of register management flip-flops and corresponding data buffer registers corresponding to the logic implemented by a MAC Bus In Data Multiplexor that interfaces to two network blocks;

FIGURE 12 is a state machine diagram corresponding to an xfr
15 machine for a system side input port for a MAC Bus In Data Multiplexor that interfaces to two network blocks;

FIGURE 13 is a state machine diagram for a request machine for network side output ports corresponding to a MAC Bus In Data Multiplexor that interfaces to two network blocks;

20 FIGURE 14 shows a set of register management flip-flops and corresponding data buffer registers corresponding to the logic implemented by a MAC Bus Out Data Multiplexor that interfaces to two

network blocks;

FIGURE 15 is a state machine diagram for an xfr machine for a system side input port for a MAC Bus Out Data Multiplexor that interfaces to two network blocks;

5 FIGURE 16 is a state machine diagram for a request machine for network side output ports corresponding to a MAC Bus Out Data Multiplexor that interfaces to two network blocks;

FIGURE 17 is a state machine diagram for an mrequest machine for a system side output port for a MAC Bus Out Message Multiplexor that
10 interfaces to two network blocks;

FIGURE 18 is a state machine diagram for an mxfr machine for A and B network side input ports;

FIGURE 19 is a timing diagram corresponding the MAC Bus In Data multiplexor during an exemplary pair of data transfers from
15 respective network side blocks to a system side block;

FIGURE 20 is a timing diagram corresponding to an operation of the MAC Bus Out Data multiplexor during an exemplary transfer of a pair of data transfers from a system side block to a respective network side blocks; and

20 FIGURE 21 is a timing diagram corresponding to an operation of the MAC Bus Out Message multiplexor during an exemplary transfer of message data between two network side blocks and a system side block.

042390.P9927

DETAILED DESCRIPTION

The present invention comprises a MAC Bus interface definition and multiplexor scheme that may be implemented to provide chip layout-insensitive connections between a number of communication physical
5 layer port entities and a single buffer manager or communications controller entity, utilizing a set of independent pipelined buses.

FIGURE 1 shows a high level view of a typical network communications interface scheme. Data passing to and from the system side is stored in a buffer 10. In order to meet a particular networks
10 message handling requirements, for the network, outbound data is sent to a MAC layer 12, which controls access to a network 14 through a physical layer 16. MAC layer 12 is also used to control access to inbound data from network 14.

Under the IEEE 802 communications standard, the MAC layer
15 comprises one of the two sublayers in the data link layer. The MAC layer controls the means by which multiple devices share the same media channel. The MAC layer also provides addressing information for communications between network devices.

FIGURE 2 illustrates a configuration in which buffer 10 is
20 connected to several MAC layer interfaces 12A, 12B, and 12C. As depicted in the Figure, buffer 10 is also connected to a memory 18 and a processor 20 that enables execution of an application program accessing

the network.

FIGURE 3 illustrates further details of the FIGURE 2 configuration in which connections to the multiple MAC layer interfaces are facilitated by a pair of MAC bus multiplexors (MUXs) 22 and 24 that control the flow of data and handshaking signals flowing across The MAC bus. In addition, the Figure illustrates the three separate bus components that make up the MAC bus interface, including MAC Out data busses 26, MAC Out message busses 28, and a MAC In data busses 30. Accordingly, as described below, each of MAC bus MUXs 22 and 24 comprise three logically separate bus multiplexors.

The MAC Bus is an intra-chip interface bus definition for unidirectional, synchronous, point-to-point data packet transfers. Its function is to support the transfer of data packets between functional blocks in communications products. Its purpose is to permit efficient movement of data associated with multiple channels and ports between media interface blocks, protocol-processing blocks, fifo management blocks and buffer management blocks. Blocks communicating across a MAC Bus interface may assume or impose additional formatting rules beyond those defined in the following definition on the data within the data packet transfers to suit the nature of their communications and processing functions.

There is a directional context for the MAC Bus interfaces. Data is

viewed to be passing across an interface where one side is closer to the network connection and the other side is closer to the local system interface. In accordance with this view, the two sides of the interface are called "System" side and "Network" side. Further, the terminology used

5 herein describes data movement in terms of data moving **In** from or **Out** to the Network. Therefore, a MAC Bus interface that moves data from a System side block to Network side block is termed a MAC Bus **Out** connection. The reverse interface, from a Network side block to System side block, is termed a MAC Bus **In** connection.

10 The handshake signal definitions used in both the MAC Bus In and Out interfaces are identical, so it is also appropriate to speak of the sending side and the receiving side of the interface. The sending side always initiates data packet transfers by asserting **data** and a

15 corresponding transfer **request** signal. The receiving side controls the actual transfers by asserting its **xfr** (transfer) signal during cycles when it latches data from the sending side. Both sides have the ability to introduce wait states so that the implementation of blocks may accommodate shared access to common buses or buffer memory in their implementation of a MAC Bus interface. The internal design of blocks

20 using MAC Bus interfaces should be such that clock cycles lost to wait states represent a minority of available cycles. Single wait states should be sufficient to permit resource sharing.

The Network side is responsible for the rate and quantity of data transfer, and also for the order of channel or port transfers that may be required to pass across an interface. As a result, the MAC Bus In interface may simply comprise a data bus and handshake signals with all requests for data transfer originating from the Network side. The design of the blocks passing data across a MAC Bus In interface should guarantee that all data transfers requested by the Network side will be accepted and completed by the System side.

The MAC Bus Out connection supports data flow from System side blocks to the Network side blocks. As the Network must determine the general timing and order of transfers, the MAC Bus Out interface must include both a *forward data transfer interface* and a *reverse request message interface* by which the network side can indicate the order and size of data transfers it requires. The reverse request message interface is similar to the forward data transfer interfaces of the In and Out connections except that it is limited to one word transfers. Those words correspond closely with the packet header words of the data transfer interfaces.

All data and handshake signals change synchronously at the end/beginning of a clock cycle. Signals need only propagate across the interface wiring from one side to the other to be latched or sampled at the end of the clock cycle with allowance for limited multiplexing and

combinatorial logic in the signal path at the interfaces. Each MAC Bus interface data transfer is initiated by the sending side with the assertion of the **request** signal together with presentation of the packet header word on the **data** bus. When the receiving side recognizes the **request** and is

5 prepared to take the data, it asserts a **xfr** signal at the start of the clock cycle. The receiving side then latches data at the end of the cycle and expects the sending side to present new data for transfer on the following cycle. If the sending side is unable to present new data at the end of the current clock cycle, it must assert **hold** during the cycle. If the receiving side detects that **hold** is true at the end of a **xfr** cycle, it must not perform another **xfr** until it detects that **hold** is false at the end of a subsequent clock cycle. If the receiving side is unable to perform a transfer in the current cycle it merely withholds **xfr** for the cycle.

The receiving side knows that the current transfer is the last word in the packet when **request** is false at the end of the cycle. It is not necessary for logic in the receiving side interface to interpret the length field of the packet header word. The sending side will remove its **request** at the end of the **xfr** cycle that transfers the next to last word of the data packet.

20 The MAC Bus interface definition is intended to perform simple multiplexing configurations with, for example, multiple interfaces on the Network side connected to a single interface on the System side by use of

MAC Bus Multiplexors. To aid in solving chip layout and routing problems, these Multiplexors may be cascaded. Data is always presented together with a new request so that the routing information in the packet header word (the port and channel fields) is present if needed by the

5 receiving interface. The routing information will be used by a MAC Bus Mux to determine on which sending side interface it should assert a request to transfer the data. Intermediate MAC Bus Multiplexors that have multiple sending side interfaces may use the routing data to avoid driving data on interfaces not selected by the routing data.

10 The MAC Bus interface signals and Header and Message word formats are shown in the following tables.

MAC Bus In data packet bus

Signal	Source	
request	network	Asserted to request transfer of data packet, removed at end of xfr cycle that transfers next to last word.
hold	network	Asserted during a cycle to indicate that network will not be able to update data output at end of current cycle, this will introduce a wait state on the next cycle if the system performs a xfr while hold is asserted.
xfr	system	Signifies that the system is latching data word at the end of the current cycle.
data[31:0]	network	32 bit data bus

MAC Bus Out message back bus

Signal	Source	
mrequest	network	Asserted to request transfer of request message from network side.
mxfr	system	Signifies that system is latching message word at the end of the current cycle.
mdata[31:0]	network	32 bit request/message bus

MAC Bus Out data packet forward bus

Signal	Source	
request	network	Asserted to request transfer of data packet, removed at end of xfr cycle that transfers next to last word.
hold	network	Asserted during a cycle to indicate that system will not be able to update data output at end of current cycle, this will introduce a wait state on the next cycle if the network performs a xfr while hold is asserted.
xfr	system	Signifies that the network is latching data word at the end of the current cycle.
data[31:0]	network	32 bit data bus

5

Header word format

4 bits	6 bits	6 bits	4 bits	12 bits
type	flags	length	port	channel

Message word format

4 bits	6 bits	6 bits	4 bits	12 bits
request	reserved	length	port	channel

Each packet transfer begins with a packet header word whose

10 format is shown above and in FIGURE 6. In addition, each message word

has a format shown above and in FIGURE 7. In each of these word formats, the flag field will indicate status and control information to be communicated with the packet. Examples are frame synchronization in T1 ports, frame status in HDLC processing and forced errors and abort in

5 frame transmission. The definition of the flag bits will be dependant on the context of the blocks using the interface.

The set of communication signals comprising the MAC Bus In interface 31 is shown in FIGURE 4. This interface supports sending data from the Network side to the System side. This operation is initiated by

10 the Network side, which asserts a **request** signal 32 and **data** 34 comprising a data packet header word and one or more 32-bit data words. The System side then controls the transfer of **data** 34 by asserting a **xfr** signal 36 during cycles when it latches the data. If necessary, a **hold** signal 38 may be asserted by the Network side during a cycle to indicate

15 that the Network side will not be able to update **data** 34 at the end of the current cycle, introducing a wait state on the next cycle if the System side performs a **xfr** signal 36 while **hold** signal 38 is asserted.

The set of MAC Bus Out interface communication signals corresponding to sending data from the System side to the Network side

20 is shown in FIGURE 5. As discussed above, sending data out requires the Network side to determine the general timing and order of transfers. Accordingly, the Mac Bus Out interface includes a forward data transfer

interface 39 comprising a **request** signal 40, a **hold** signal 42, a **xfr** signal 44, and **data** 46, as well as a reverse request message interface 41 comprising an **mrequest** signal 48, an **mxfr** signal 50, and an **mdata** message word 52. A data transfer out operation is initiated by the System side by asserting **request** signal 40 and **data** 46. The Network side then controls transfer of **data** 42 by asserting **xfr** signal 44. In a manner similar to MAC Bus In interface 31, **hold** signal 42 may be asserted by the System side during a cycle to indicate that the System side will not be able to update **data** 46 at the end of the current cycle, introducing a wait state on the next cycle if the System side performs a **xfr** signal 44 while **hold** signal 42 is asserted.

Further communications control is enabled through the data request messages passed from the Network side to the System side over reverse request message interface 41. These messages are passed in a manner substantially similar to that discussed above with respect to the MAC Bus In interface 31, except that the message data is moving in the opposite direction, and is limited to one word transfers (there is no equivalent to hold signal 38), wherein **mrequest** 48 is asserted to request transfer of a request message from the Network side while asserting a **mdata** message word 52 and **mxfr** signal 50 is asserted to indicate that the System side is latching **mdata** message word 52 at the end of the current cycle.

In order to simultaneously send data to and receive data from multiple network channels, one or more MAC Bus multiplexor are needed. Each MAC Bus multiplexor comprises three components: a MAC BUS In multiplexor, a MAC Bus Out Data multiplexor, and a MAC Bus Out

5 Message multiplexor.

An exemplary MAC Bus In Data multiplexor (Mux) 54 is shown in FIGURE 8. MAC Bus In Data Mux 54 provides two-to-one multiplexing to facilitate inbound communications between a System side block 56, and two Network side blocks A and B. Accordingly, the MAC Bus In interfaces

10 between the Network side input ports and MAC Bus In Data Mux 54 are labeled "31A" (block A) and "31B" (block b), while the MAC Bus In interface between MAC Bus In Data Mux 54 and System side block 56 is labeled "31SS" (system side).

MAC Bus In Data Mux 54 includes a control 58, a pair of 32-bit

15 multiplexors 61 and 62, and buffers 64 and 66. As described below, buffers 64 and 66 preferably comprise registers, although similar components used for temporarily holding data may also be used. Control 58 is used to control the multiplexing of the various signals received by and asserted from the Network side input ports and the

20 System side block. Control 58 multiplexes **request** and **hold** signals received from Network side input ports A and B, and forwards these signals with appropriate timing to System side block 56. In addition,

Control 58 receives **xfr** signals from System side block 56, and multiplexes these signals with appropriate timing to input ports of Network side blocks A and B.

In addition to controlling forwarding of the foregoing signals, control 58 controls the flow of **data 34** across the interface by asserting data movement control signals 68, 70, 72, 74 to respectively control multiplexor 61, buffer 64, buffer 66, and multiplexor 62. The use of dual data buffers permits the sustained transfer of data packet words through the Mux from the Network side blocks to the System side block. The timing of the assertion of the data movement control signals and forwarding the **request**, **hold** and **xfr** signals is controlled by a corresponding state machine. State machine diagrams and a timing diagram corresponding to the configuration shown in FIGURE 8 are respectively shown in FIGURES 11, 12, and 19.

An exemplary MAC Bus Out Data Mux 76 is shown in FIGURE 9. MAC Bus Out Data Mux 76 comprises a pair of buffers 78 and 80, and a pair of multiplexors 82 and 84, that receive data movement control signals 86, 88, 90, and 92 from a control 59. Control 59 receives **request** and **hold** signals issued by System side Application 56, multiplexes these signals, and forwards them with appropriate timing to Network side block A and B, while receiving **xfr** signals from the Network side blocks and forwarding the **xfr** signals to System side block 56 with appropriate timing.

Data 46 are sent from System side block 56 to Network side blocks A and B via buffers 78 and 80 and multiplexors 82 and 84. As with control 58, control 59 controls the delivery of the data words based on the handshaking signals it receives. In addition, routing data bits 94 are

5 extracted from data headers words corresponding to data 46 in buffers 78 and 80, and provided to control 59 to enable the control to route data 46 to the appropriate System side input port. The use of dual data buffers in the Mux permit the sustained transfer of data packet words through the Mux from System side block 56 to the Network side blocks. State

10 machine diagrams and a timing diagram corresponding to the Mux configuration shown in the FIGURE 9 are respectively shown in FIGURES 15, 16, and 20.

FIGURE 10 shows an exemplary MAC Bus Out Message Mux 96 that includes a single multiplexor 98, which is controlled by a control 60 via a "selA" control signal 100. Control 58 receives **mrequestA** and

15 **mrequestB** signals 48A and 48 B from the Network side blocks A and B, multiplexes the requests, and passes them on to System side block 56 with appropriate timing. Control 60 also receives **mxfr** signals from the System side block, multiplexes the signals, and deliverers the **mxfr**

20 signals to Network side blocks A and B with appropriate timing.

MAC Bus Out Message Mux 96 relies on the Network side to hold **mdata** stable so that an additional data latch in the Mux is not needed.

Data propagates from the Network side to the Mux in one cycle, and through the Mux to the System side during a subsequent cycle when the Mux asserts its **mrequest** signal to the System side interface. Once the System side **mxfr** is seen by the Mux, **mxfr** is presented to the Network
5 side to confirm the message transfer.

In the foregoing discussion, each of Mac Bus In Data Mux 54, Mac Bus Out Data Mux 76, and MAC Bus Out Message Mux 96 comprise separate components. This configuration provides the benefit that each Mux may be operated with a separate clocking signal. Optionally, controls
10 58, 59, and 60 may comprise a single control that uses a single clocking signal for all three bus muxing functions.

Logic for implementing Mac Bus In Data Mux 54 is now presented with reference to FIGURES 11, 12, and 13. FIGURE 11 shows a set of register management flip-flops that may be used to implement the logic for
15 controlling the data buffer registers (i.e., buffers 78 and 80). The reg#full bits are set when an xfrA or xfrB to the register takes place and are cleared when an xfr from the register takes place. The input is 1 bit toggles on each xfrA or xfrB to either reg1 or reg2. The output is 1 bit toggles on each xfr from either reg1 or reg2. The reg#last bits are set
20 when the associated register is loaded within a transaction and the request input for the transaction is low. They are cleared when the output xfr from the register is completed.

FIGURE 12 shows a machine state diagram corresponding to data transfers from Network side blocks A and B, wherein the states for each block are indicated by an appended "A" or "B." A legend for the various states is shown in the lower left-hand corner of the Figure. An IdleIn state 102 is shared by both transactions, wherein an input port A transaction further includes an xfrA state 104, a waitA state 106, and a fullA state 108, and an input port B transaction further includes an xfrB state 110, a waitB state 112, and a fullA state 114. Requests and hold signals that are issued to cause the machine to move to the next state are depicted adjacent to the arrows connecting each state. The state machine also includes recursive loops, as shown by nearly complete circles with an arrow on one end.

FIGURE 13 depicts a request state machine for a System side block output port, which includes an IdleOut state 116 and a req-out state 118.. "Output-islast" indicates the current output register's reglast bit. On the last xfr of a transaction, if both registers are full, the logic remains in req-out state 118, and another transaction is ready.

The hold output may be produced combinatorially from the state of the data buffers and the xfr machine as follows:

$$\text{hold} = (\text{full} + \text{xfrA} + \text{xfrB})^*$$

The value of hold is a don't care on the last xfr of a message. If hold delays the last xfr, it must go low for at least one cycle to permit it to

occur.

FIGURES 14, 15, and 16 depict MAC Bus Out Data Mux logic.

With respect to the register logic of FIGURE 14, the reg#full bits are set when an xfr to the register takes place and are cleared when an xfrA or xfrB from the register occurs. The inputis1 bit toggles on each xfr to either reg1 or reg2. The outputis1 bit toggles on each xfrA or xfrB from either reg1 or reg2. The register's route2A is loaded based on routing information in the message header during a first-xfr. For other xfr's, the inputis register's route2A bit receives the other register's route2A value.

10 The reg#last bits are set when the associated register is loaded within a transaction and the request input for the transaction is low. They are cleared when the output xfr for the register is completed.

A xfr state machine for a System side input port is shown in FIGURE 15. The states include an IdleIn state 120, a first xfr state 122, and xfr state 124, a wait state 126, and a full state 128. Events that cause transistion between the state are depicted adjacent to the arrows connecting the states, as before. A legend for the states and transitions of the xfr state machine is shown just below the diagram.

A request state machine for the Network side output ports is shown in FIGURE 16, comprising an IdleOut state 130 and a req-out state 132.

"Output-islast" is the current output register's reglast bit. On the last xfr of a transaction, if both registers are full, the logic stays in req-out state 132,

and another transaction is ready.

The hold output may be produced combinatorially from the state of the data buffers and the xfr machine as follows:

$$\text{Hold} = (\text{full} + \text{xfr})^*$$

- 5 The value of hold is a “don’t care” on the last xfr of a message. If hold delays the last xfr, it must go low for at least one cycle to permit it to occur.

State machine logic diagrams for a MAC Bus Out message Mux that interfaces to two Network I/O ports are shown in FIGURES 17 and 10 18. FIGURE 17 depicts an mrequest state machine for a System side output port, which includes an IdleIn state 134, an awaitA state 136, an awaitB state 138, and an end state 140. FIGURE 18 shows mxfr state machine diagram for the A and B Network side input port, wherein the A input port states include an IdleA state 142, and an mxfrA state 144, and 15 the B input port states include an idleB state 146 and an mxfrB state 148. Events that cause transitions between the state are shown adjacent to the arrows in the Figures.

An exemplary timing diagram corresponding to MAC Bus In Data Mux 54 is shown in FIGURE 19. As illustrated, request and data signals 20 are asserted at the rising edge of a clock cycle, and corresponding xfr signals are asserted at the rising edge of the subsequent clock cycle. The timing diagram of FIGURE 19 illustrates several timing conditions

corresponding to a transfer of data from Network side blocks A and B.

When holdA is asserted at a point 150, it has no effect, since the registers reg1 and reg2 were filled by a current xfr. At a point 152, an xfr is

permitted, but the receiver abstains. A hold at a point 154 causes xfrB to

5 become low at a point 156, while holdB must go low at a point 158 if leaving it asserted would delay the last xfr.

In the exemplary timing diagram of FIGURE 19 data from Network Side block A is received by System Side block 56 prior to receiving data from Network Side block B. In addition to this timing, other timing

10 configurations may also be implemented, wherein the logic may be designed such that a short set of data packets might interrupt a longer set of data packets.

A similar exemplary timing diagram corresponding to Mac Bus Out Data Mux 76 is shown in FIGURE 20. This diagram corresponds to a

15 system side block sending a set of data packets to Network side blocks A and B in sequence. As discussed above, the header word of a set of data packets contains information concerning what port and channel the message needs to be routed to. This information is read in by control 59, which then routes the data packets to an appropriate Network side port

20 and channel, as indicated by 160 and 162. In addition, asserting a hold at a point 164 during a first timing cycle causes holdA to be asserted at the next (second) timing cycle (164), which further causes xfr to be low (166)

during the second timing cycle and xfrA to be low (170) during a third timing cycle.

FIGURE 21 shows an exemplary timing diagram corresponding to Mac Bus Out Message Mux 96. As discussed above, MAC Bus Out Messages are a single word in length, and do not include the use of a hold signal. As a result, mrequest does not have to be removed at a point 172. Also illustrated are the one-cycle delays to mxfrA and mxfrB signals in response to asserting mxfr, as indicted by 174 and 176.

In the foregoing timing diagrams, data is transferred sequentially based on an order in which data sending requests were issued by Network side blocks A and B. As discussed above, several multiplexors may be cascaded to enable communication between a single System side block and four or more network side blocks. In addition, the configurations shown and discussed above for MAC bus multiplexors 54, 76, and 96 are exemplary instances of 2-to-1 multiplexing. As will be recognized by those skilled in the art, many-to-one (e.g., 3-to-1, 4-to 1) may be also be implemented. Accordingly, when a multiplexer receives multiple data transfer requests, the request will need to be serviced in some order or prioritization. Preferably, these requests should be service using a round robin prioritization scheme, although other prioritization schemes may also be employed.

The above description of illustrated embodiments of the invention is not intended to be exhaustive or to limit the invention to the precise forms disclosed. While specific embodiments of, and examples for, the invention are described herein for illustrative purposes, various equivalent
5 modifications are possible within the scope of the invention, as those skilled in the relevant art will recognize. Accordingly, it is not intended that the scope of the invention in any way be limited by the above description, but instead be determined entirely by reference to the claims that follow.